REMARKS

Claims 1, 4-21 and 31-40 are pending. Claims 1, 20, 21 and 31 are amended to more particularly point out the distinctions over the cited art. Claims 32-40 are withdrawn.

35 U.S.C. § 103 Rejection

Claims 1, 4-21 and 31 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Scheifler et al. (US Patent 6,138,238) in view of Colburn et al. (US Patent 6,173,404). Applicants respectfully traverse this rejection.

Applicants respectfully submit that Scheifler and Colburn are being read too broadly. The broadest reasonable construction is one "read in the appropriate context of the claim language and the specification." *In re Suitco Surface, Inc.*, 2009-1418 (Fed. Cir. 2010). Scheifler is directed towards a security approach that utilizes a centralized security policy file to store permissions for a particular resource. The policy file interacts with the call stack and an execution unit. The call stack contains invocation of methods. The execution unit grants access to the resource "when the types of access authorized by the permissions of all of the methods and executors on the call stack encompass the access requested by the operation." Col. 4, line 57 – col. 5, line 3. The examiner is reading Scheifler's capabilities to be outside this explicit definition. To previous responses, the examiner has stated that the examiner is not clear where in Scheifler that security detail is centralized, where Colburn discloses authorization that creators must implement into their objects, and that Schiefler and Colburn can be combined simply because they both use object oriented programming concepts. Applicants disagree.

Scheifler's and Colburn's security paradigms are based on permissions being granted based on source, executor, or owner of objects. While some of their security features are incorporated into objects, the structure and functionality of Scheifler and Colburn, alone or in combination, are different than the claimed invention.

Scheifler's Security Scheme

Scheifler, Figure 4, is a representation of Scheifler's system that exemplifies a "centralized" security scheme. Security (e.g., permissions) is stored in a policy file (Fig. 4, ref. 4100). This

policy file is the centralized storage for the various permissions available within the resource (though the word "centralized" is not used, the structure of Scheifler when properly examined reveals the centralized structure). Each permission, executor, resource, and capability are represented in this file (See e.g., Fig. 5, which stores permissions for Executor 1 to Executor N.). The security mechanism of a system in Scheifler uses "permission objects and protection domain objects to store information that models the security policy of the system." Col. 8, lines 24-27. A protection domain is "a set of permissions granted to one or more executors when code from one or more sources is being executed on their behalf." Col. 11, lines 23-26. As shown in Fig. 4, the protection domain object (ref. 4400) is external to object (ref. 4500). A permission object may also represent permissions of a system. Permission objects derive permissions from the policy file (ref. 4100) for a given system. See Fig. 4. The permission object contains the methods for determining permissions of other objects. Col. 11, lines 54-56. Any security permission and access (whether implied or explicit) is validated using a method within a permission object. See e.g., col. 12, lines 46-55.

To implement the security policy of the system, a policy object, domain mapper object, one or more protection domain objects, and one or more access identifiers are needed. Col. 12, lines 61-65 and Fig. 4. Scheifler outlines the centralized structure of the system:

> Policy object 4200 is an object for storing the policy information obtained, for example, from policy file 4100. Specifically, policy object 4200 provides a mapping of access identifiers to permissions, and is constructed based on the instructions within policy file 4100. Within the policy object 4200, the access identifiers and their associated authorized permissions may be represented by data structures or objects. Col. 12, line 66 - col. 13, line 6.

To simplify, Schiefler stores security policy that is centrally designated by a policy file in a policy object. This policy object is external to an object that may utilize its security validation methods.

A protection domain object is created when new access identifiers (e.g., new executors of code) are encountered. The protection domain object gets populated with the permissions for a particular executor that are obtained from the central policy file through the policy object. See col 13, lines 7-30. Even in alternative implementations, the mapping of permissions from the policy file to a protection domain class is stored in the protection domain class. Col. 13, lines 30-41. These access permission are not resident within a target object as claimed.

Actions are authorized if such a permission is incorporated within a protection domain object within a thread (representing a major structural difference between Scheifler and the present invention as claimed). Scheifler states:

> According to an implementation consistent with the present invention, an action is authorized if the permission required to perform the action is included in each protection domain object associated with the thread at the time when a request to determine the authorization is made. A permission is said to be included in a protection domain object if that permission is encompassed by one or more permissions associated with the protection domain object. For example, if an action requires permission to write to a file in the "e:/tmp" directory on behalf of the principal "Bob," then that required permission would be included in protection domain object 4400-1 if the protection domain object 4400-1 explicitly contains or implies that permission.
>
> Assume that thread 6200 is executing method 6300-3 when thread 6200 makes a request for a determination of whether an action is authorized by invoking the check permission method 6400. Assume further that thread 6200 has invoked method 6300-1, method 6300-2, and method 6300-3 and these methods have not completed when thread 6200 invoked method 6400. The protection domain objects associated with thread 6200 when the request for a determination of authorization is made are represented by protection domain objects 4400-1, 4400-2, and 4400-3.
>
> Given the calling hierarchy present in the current example, the required permission to perform an action of writing to file "d:/sys/pwd" on behalf of "Bob" is not authorized for thread 6200 because the required permission is not encompassed by any permission included in protection domain object 4400-1, if the only permission contained therein is "write to e:/tmp." Col. 15, lines 25-55.

So, permissions in Scheifler are not determined at an interface of the target object as presently claimed, but by consulting a protection domain object that derives its permissions from a policy file that is centralized with respect to the policy objects, domains, and executing objects. When Scheifler and the present invention as claimed are read within their proper contexts, Scheifler's system and method present a differing structure and functionality that is distinguishable from the presently claimed invention. That they both utilize object oriented programming techniques to implement their differing structures and method is irrelevant (It is akin to stating that because two hardware inventions incorporate silicon, then they are relevant to combine.). Because both may utilize object oriented programming is an improper reason for rendering the claims obvious or combining two references.

Colburn's Security Scheme

Colburn's inter-object security scheme requires an owner identifier incorporated into the objects.

> The owner identifier includes identification of the user, person, or entity (e.g., corporation) who or that creates the object, or identification of a computer system used by the user, person, or entity to create the object definition. The owner identifier provides a basis for distinguishing the creator of an object from the user of that object. This distinction allows instances of objects created by others to be given fewer access permissions or rights than the user implementing the object. Col. 1, lines 55 – 63.

To resolve the security status of an owner identifier, Colburn requires identification of the entity that creates an object definition and access is granted with regard to the computer system. Security permissions are not granted based on a call to a first interface and the security policy of a target object is certainly not contained solely within the target object as claimed. Colburn states

> Exemplary owner identifiers include identification of the user, person, or entity (e.g., corporation) who or that creates the object definition, or identification of a computer system used by the user, person, or entity to create the object definition. The object identifier may be in the form, for example, of a pointer to the owner IThing COM object. For purposes of illustration, process 180 is described as if the Owner identifier references the individual user who creates the object definition. It will be appreciated, however, that the Owner identifier could alternatively identify an entity (e.g., corporation or educational institution) where or a computer system on which the object definition is created.
>
> Process block 190 indicates that the object is made available for use, either by the user who created the object or one or more other users.
>
> Process block 192 indicates that a selected user instantiates on the selected user's computer system an accessing instance of an object (e.g., accessing instance 156) based upon the object (e.g., object 154).
>
> The accessing object will seek access to a target (e.g., a method or property) of a target instance (e.g., target instance 164) on the selected user's computer system. The selected user has a set of permissions or rights with regard to the computer system that allow the user to access a wider range of computer system resources than can be accessed by other users, particularly users who are unknown to the system. In addition, different classes of target object services have different access authorizations (e.g., access authorizations 194, FIG. 8) that represent different permissions or rights for performing different access operations. The

> target access authorizations indicate which objects are allowed access to particular services on a target object.
>
> In one implementation, three levels of access authorizations 194 are All, Owner, and Exemplar. Object services or targets with the access authorization All indicate that all objects, regardless or their Owner, may perform access operations on the targets. Targets with the access authorization Owner indicate that only the owner of the target instance may access it. Object services with the access authorization Exemplar indicate that only services defined on the same exemplar as the target instance may access those targets. In this implementation, the Exemplar access authorization is the most restrictive and the All access authorization is the least restrictive. It will be appreciated, however, that other access authorization levels could be utilized, relating to a variety of considerations. Col. 10, line 60 – col. 11, line 38.

Access is granted based on a combination of owner identifier and access authorizations. The access authorizations are not interface based but arbitrary designations that enable different levels of access to the objects. The access authorizations are dependent on whether a particular object belongs to an object security class. See e.g., Colburn, claim 1 ("An access authorization security condition associated with the service and conditioning access to the service by the second object on whether the second object is in one of plural object security classes, one of which being a class in which the first and second objects are defined by the same person or entity."). In order for the access to be resolved the owner identifier must be resolved, which necessarily involves a process outside of a particular object. For example, Colburn states

> Process block 260 indicates that the server restricts the user (and the client computer) to targets having access authorizations of All.....Process block 262 indicates that accessing objects from the user (and the client computer) are permitted to access targets on the server having Owner access authorization and in which the user is the owner. This access is permitted through targets on a call stack on the server. Col. 14, lines 5 – 13.

Moreover, Colburn's discussion of dynamic inheritance is further evidence that security is not determined as claimed. See e.g., Fig. 11 and corresponding description col. 14, line 35 – col.16, line 21.

Scheifler-Colburn Combination is Improper

Combining Colburn with Scheifler impermissibly changes the principle operation of Scheifler. Scheifler stores security details (e.g., permissions) in a centralized policy file not in

target objects as claimed. See e.g., Figure 4. Permissions authorizing access are based on source and executor of a piece of particular code. Scheifler explicitly states

> A security enforcement mechanism is provided in which the access permissions of a thread are allowed to vary over times based on the source and executor of the code currently being executed. The source of the code indicates whether the code is from a trusted or untrusted source. The executor indicates the principal on whose behalf the code is being executed. For example, the executor may be a particular user or a particular organization on whose behalf the process or program is operating on a client computer. . . . According to an implementation consistent with the present invention, the security mechanism described herein uses permission objects and protection domain objects to store information that models the security policy of a system. The nature and use of these objects, as well as the techniques for dynamically determining the time-variant access privileges of a thread, are described hereafter in great detail. Col. 7, line 65 – col. 8, line 30.

Colburn relies on an owner-identifier being incorporated into objects. This identifier is based on the creator of an object or the system used to create the object. So that Colburn's system may function, Colburn defines a set of access authorizations that creators must implement into their objects. See col. 9, Table 1, Attribute Name Access Authorization (Applicants believe this table along with the associated discussion below the table, col. 9, lines 21-41, outlines the definitions of the set of access authorization clearly enough so that the examiner may "properly respond to applicant's arguments."). Colburn's system is not based on a centralized authority controlling security details, but the existence of an owner identifier and a standardized system of access authorizations. See e.g., Abstract, col. 8, line 60 – col. 9, line 41, and col. 10, lines 6 – 51. Combination of Colburn with Scheifler requires abandoning Scheifler's use of a centralized authority (e.g., the policy file) to determine security. Conversely, incorporating Scheifler into Colburn requires Colburn to adopt Scheifler's use of centralized permission objects. Both systems describe two different specific implementations of controlling access that are not compatible. For this reason alone, the combination is improper and the prima facie case of obviousness has not been met.

However, even a combination does not teach or suggest the claimed invention. Scheifler's disclosure of implied permissions to other interfaces does not read upon the present claims. Therefore, implied permissions in the combination of Scheifler and Colburn cannot read upon the present claims regardless of the implementation. The present claims do recite that

permissions are implied as described in Scheifler. As stated, interface permissions in the present invention as claimed are not implied. Each interface may grant varying degrees of access to the target object. See e.g., Specification para. [0058] (such varying degree of access implied by the claim language "determining whether the external object has access to other interface of the target object based on the call to the first interface"). Scheifler's disclosure of implied permission does not constitute determining access to other interfaces of a target object as the examiner implies. Scheifler explicitly states:

> If a permission is represented by a permission object, the validation method for the permission object contains code for determining whether one permission is implied by another. For example, a permission to write to any file in a directory implies a permission to write to any specific file in that directory, and a permission to read from any file in a directory implies a permission to read from any specific file in that directory. However, a permission to write does not imply a permission to read. Col. 12, lines 46-55.

In the present invention, access to one interface does not "imply" access to another interface. See, e.g., Specification, para. [0056] ("Each of these interfaces grants a specific set of permissions to any object obtaining a reference to it..."). As Scheifler explicitly states, the permission object contains code for determining whether one permission is implied by another. The present invention as claimed does not determine whether a permission is implied based on another permission. Rather the target object determines whether an external object access to a particular interface based on a call to the first interface. See e.g., Specification, para. [0058].

Adding Colburn to Scheifler does not solve either's deficiencies. Scheifler's security model requires a centralized authority to determine permissions, which does not read on the claims. Incorporation of Colburn impermissibly requires modifying the principle of operation of Scheifler, or in the alternative, the combination still requires a centralize authority to determine permissions. In either case, neither reference discloses permissions based on the call to the first interface as claimed. Scheifler and Colburn, individually and in combination, do not teach or suggest each and every element of the claims. Accordingly, the Applicants respectfully request withdrawal of this rejection.

Conclusion

All of the stated grounds of rejection have been properly addressed. Applicants therefore respectfully request that the examiner reconsider the outstanding rejections and allow the present claims. The examiner is invited to telephone the undersigned representative if an interview might expedite allowance of this application.

Respectfully submitted,

BERRY& ASSOCIATES P.C.

Dated: July 25, 2011

By: /Shawn Diedtrich/
Shawn Diedtrich
Registration No. 58,176
Direct: 480.704.4615

9229 Sunset Blvd., Suite 630
Los Angeles, CA 90069
(310) 247-2860